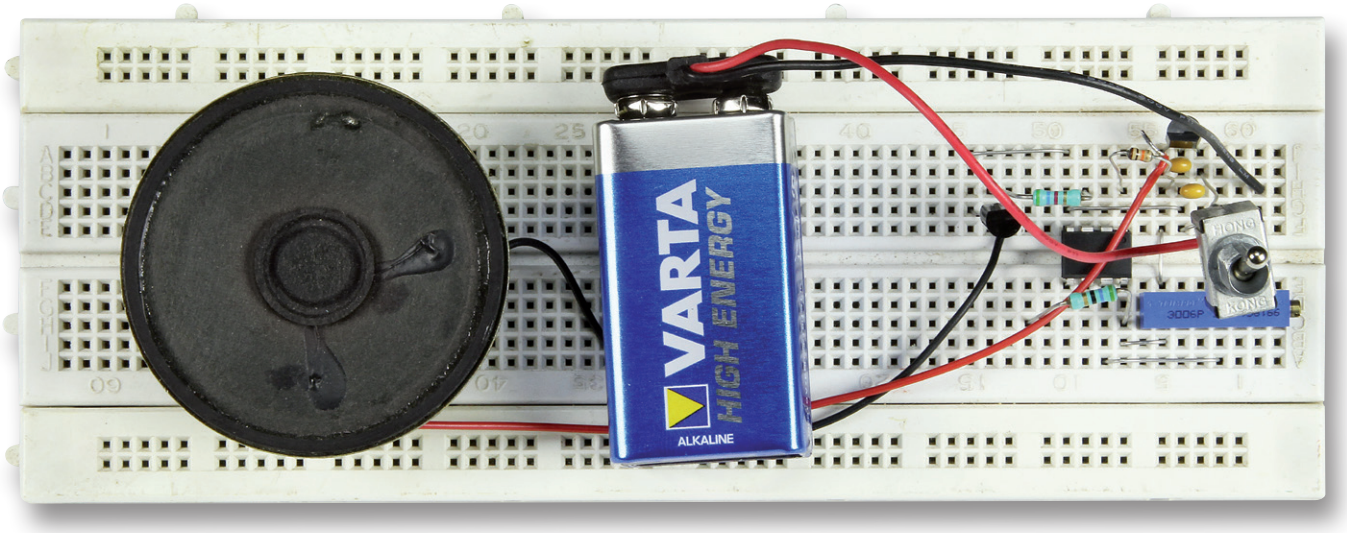
bruiteur 2.0

Tintouin et l'oreille cassée

Friedrich Lischeck
(Allemagne)

À l'époque où fut publié le schéma du bruiteur, en 1979, l'auteur pouvait à juste titre s'enorgueillir de n'avoir eu besoin pour son circuit que d'un compteur CD4040, d'un inverseur CD4049, et de quelques tâcherons associés. L'électronique a évolué et nous pouvons aujourd'hui faire du boucan avec encore moins de composants. C'est aussi ça le progrès !



Impossible de ne pas le remarquer sur le schéma (**fig. 1**) : le compteur et l'inverseur d'autrefois [0] ont laissé place à un petit microcontrôleur très actuel. Cet ATtiny45 produit le signal et se charge à lui seul de l'ancienne fonction d'oscillateur à rétroaction. Autre temps, autre technique, l'astucieuse rétroaction matérielle est donc remplacée ici par un programme. Un seul transistor suffit à amplifier les signaux acoustiques. Leur sortie est dirigée vers un petit haut-parleur dynamique de 8 Ω, et la fréquence fondamentale du circuit s'ajuste à l'aide d'un potentiomètre.

Programme

L'intelligence de l'électronique actuelle réside plus dans les micrologiciels que dans le câblage des composants, aussi ingénieux soit-il. De nos jours, il semble judicieux de confier au µC toutes les tâches d'un circuit, mêmes les plus simples. Le programme du contrôleur AVR utilisé ici (voir **encadré**) a été écrit en BASIC BASCOM [1]. Le bruiteur 1.0 d'il y a 34 ans produisait un son de hauteur croissante qui cessait une fois la fréquence maximale atteinte, puis repartait brusquement depuis la fréquence de départ. La vitesse d'augmentation dépendait du réglage de la fréquence initiale.

Ce même signal est aujourd'hui produit par l'ATtiny45. Le potentiomètre permet d'appliquer sur l'entrée analogique ADC.1 une tension comprise entre 0 et 5 V. Le CAN de l'ATtiny convertit cette tension en une valeur numérique comprise entre 0 et 1023. Cette valeur sert ensuite à initialiser la fréquence `Fg`. Le comportement du temporisateur `Timer0` en dépend.

Une interruption est déclenchée à chaque débordement de `Timer0`. Le temporisateur est ensuite initialisé avec une nouvelle valeur de départ dans la routine d'interruption `Oscillateur`, puis l'état de la sortie numérique B.4 est basculé. Le résultat est un signal rectangulaire symétrique. La variable `Fd` est en outre incrémentée jusqu'à ce qu'elle dépasse 1000 ; on lui affecte alors de nouveau la valeur 100. Enfin le programme principal fusionne les deux fréquences : `Fd` est additionnée à `Fg`, c'est-à-dire à la valeur qui dépend du réglage du potentiomètre. La fréquence résultante est comprise entre 200 Hz et 4 kHz. Elle augmente sur un intervalle de 1000 Hz, puis le piaulement reprend avec la valeur initiale plus basse.

Valeur initiale

Voyons comment est calculée la valeur initiale de préchargement `Preload` du temporisateur. Comme vous le savez sans doute, le temps `T` (durée entre 2 interruptions) d'un temporisateur à 8 bits est défini par :

```
T = Tmax – Preload * Tclk
```

La valeur `Preload` doit être comprise entre 0 et 255. `Tclk` est la période de l'horloge du contrôleur, valeur qui le cas échéant peut dépendre du réglage du prédiviseur (1/8/64/256). `Tmax` est le temps maximal du temporisateur pouvant être paramétré :

```
Tmax = Prescale * 256 / f
```

où f est la fréquence d'horloge. La première équation peut être réécrite :

```
Preload = (Tmax – T) / Tclk
```

soit :

```
Preload = (Tmax – 1 / 2 * F) / Tclk
```

où `F` est la fréquence à produire, et `T` le temps correspondant, soit la durée de la moitié de la période.

Conclusion

L'ATtiny45 peut sembler disproportionné par rapport à la brièveté du code, mais c'est le modèle que j'avais sous la main. Vous pouvez utiliser un ATtiny25 et sa mémoire flash de 2 Ko. La version de démonstration de Bascom-AVR suffit pour compiler le programme puisqu'elle accepte jusqu'à 4 Ko de code. Que vous vous serviez d'un programmateur acheté ou assemblé, pensez à deux choses au moment d'écrire le code dans le microcontrôleur : d'abord à sélectionner le bon modèle de µcontrôleur, ensuite à paramétrer les fusibles. Sur la **figure 2** (menu *Fuses* du logiciel Atmel Studio), vous noterez que le fusible de division interne de l'horloge est désactivé, et que l'oscillateur interne est à 8 MHz avec un temps de départ long (fusible CKSEL). La présence d'erreurs entraîne en général une fréquence trop basse ou une

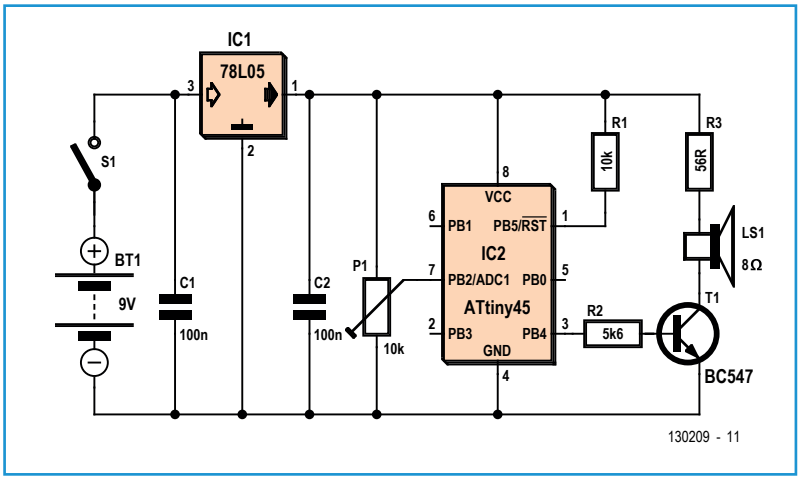
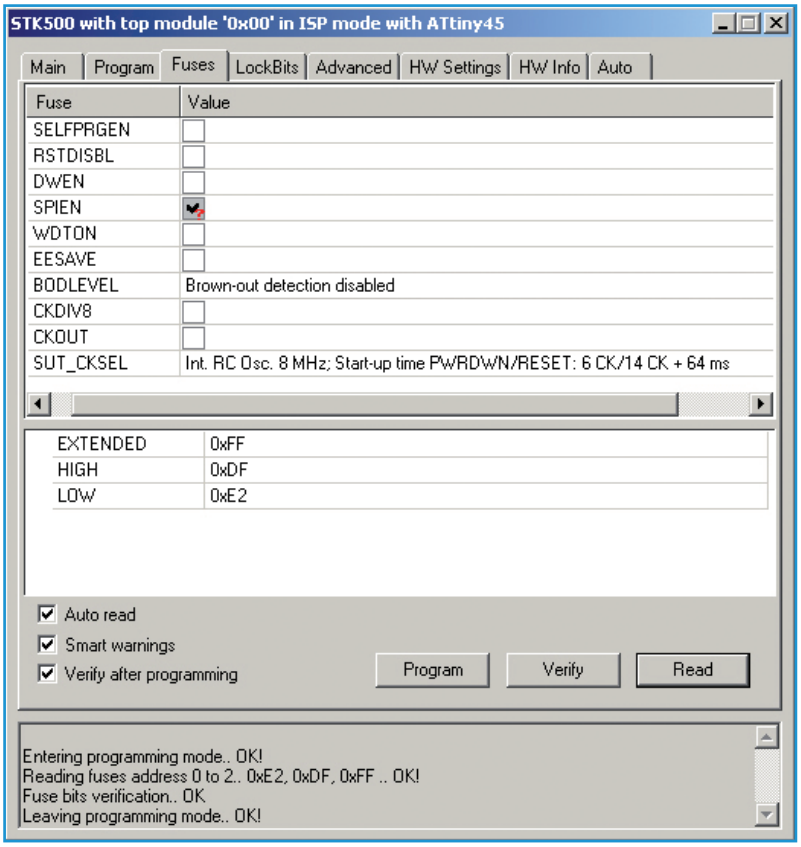
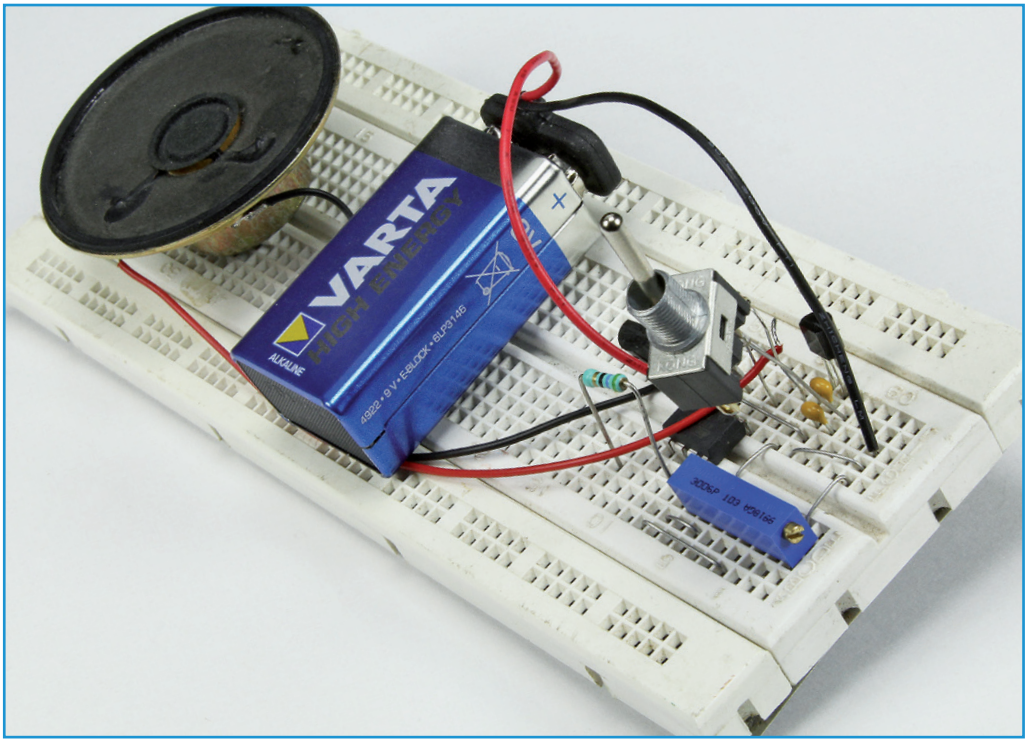


Figure 1.
Le circuit du bruiteur 2.0 est encore plus simple que celui de 1979.

Figure 2.
Paramétrage des fusibles de l'ATtiny45.





Liste des composants

Résistances
(toutes ¼ W)
R1 = 10 kΩ
R2 = 5,6 kΩ
R3 = 56 Ω
P1 = pot. 10 kΩ

Condensateurs
C1, C2 =
100 nF, 16 V, cér.

Semi-conducteurs
T1 = BC547
IC1 = 78L05
IC2 = ATtiny45, programmé

Divers
Support DIL à 8 broches pour IC2
Coupleur à pression pour pile de 9 V
B = pile de 9 V
S1 = interrupteur
LS = haut-parleur, 8 Ω / 0,5 W

absence de son. Vous pouvez télécharger gratuitement le fichier binaire et le code source depuis [2].

Peu à dire sur le montage. Vous n'aurez aucune peine à insérer les quelques composants sur un morceau de plaque d'essai. Un support DIL est recommandé pour IC2. Terminons avec cette phrase historique, qui ouvrait l'article de septembre 1979 : « Nous avons quelque part dans les laboratoires d'Elektor, un département des effets sonores [...]. Ce que l'on entend dans ce département ressemble à des cris perçants de chats torturés agonisant, à des hurlements horrifiants et à tout un assortiment de plops, bangs sifflements... ». En 2013, ça n'a guère changé.

(130209 – version française : Hervé Moreau)

Liens

[0] générateur simple de sons bizarres
Elektor nº15, septembre 1979, page 38
[1] Bascom-AVR : www.mcselec.com
[2] www.elektor-magazine.fr/130209

Le code

```
,bruiteur avec Tiny45
$regfile = «attiny45.dat»
$crystal = 8000000

,broche B.0 configurée en sortie :
Ddrb = &B00010000
'broche 4 à l'état 0 :
Portb.4 = 0

Dim Preload As Byte
Dim F As Word
```

```
Dim Fg As Word
Dim Fd As Word
Dim Tmax As Single
Dim Tclk As Single
Dim H As Single

On Timer0 Oscillateur
Config Timer0 = Timer , Prescale = 256
Enable Timer0
Enable Interrupts
Preload = &H64:                                    ,100 Hz
Timer0 = Preload

Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc
,-------------------------------------------------------
'Initialisation
Tclk = 256 / 8000000
Tmax = 256 * Tclk
Fg = 100
Fd = 100
,-------------------------------------------------------
'Programme principal
,F = 200 - 4000
Do
   Fg = Getadc(1)
   H = Fg + 100
   Fg = Fg + H
   Fg = Fg + H
   F = Fg + Fd
   H = 2 * F
   H = 1 / H
   H = Tmax - H
   H = H / Tclk
   Preload = Int(h)
   H = Preload * Tclk
   H = Tmax - H
   H = 0.5 / H
   F = Int(h)
Loop
End
,-------------------------------------------------------
Oscillateur:
   Timer0 = Preload
   Portb.4 = Not Portb.4
   Incr Fd
   If Fd > 1000 Then Fd = 100
Return
,-------------------------------------------------------
```